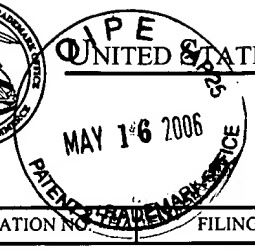
2191



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/985,880	11/06/2001	Andrew Hamilton	003636.0131	4508

7590 05/05/2006

ASHOK K. MANNAVA
281 MURTHA STREET
ALEXANDRIA, VA 22304

EXAMINER	
VO, TED T	

ART UNIT	PAPER NUMBER
2191	

DATE MAILED: 05/05/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary	Application No.	Applicant(s)	
	09/985,880	HAMILTON ET AL.	
	Examiner	Art Unit	
	Ted T. Vo	2191	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *02 December 2005*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-36* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☒ Claim(s) *4-12,16-24 and 28-36* is/are allowed.

6)☒ Claim(s) *1-3,13-15 and 25-27* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application (PTO-152)
6) ☐ Other: _____.

DETAILED ACTION

1. This action is in response to the amendment filed on 12/02/2005 entered by filing RCE on 02/21/06.

Claims 4-12,16-24 and 28-36 stand allowed.

Claims 1-36 remain pending in the application.

Response to Amendment

2. Applicants' amendment and arguments have been fully considered.

Regard to Applicant's arguments as pointing to MPEP 2131 and quotes of particularly case laws. Examiner recognized that Applicants' pointing is improper. In this reply, Applicants fail to point out the novelty presenting in the claims. The claims do not direct toward a particularly function, but merely recites many terms that can cover many aspects of the arts. Given typical examples:

"application" could be any executable file in a computer;

"run-time engine" can been read by a processor, by an operating system such as Windows, or a development toolkit, etc;

"Go method" can be read by a button click in any popup windows;

"Screen definition" is read by default setup of any image seen in computer Windows or of the Windows itself.

It should be noted that it is not necessary for a reference to document a feature that is already seen or included, common and known to a skilled artisan. Vie versa, it is not necessary for a patent application to document a well-known feature (e.g., Hybritech, Inc. v. Monoclonal Antibodies, Inc., 802 F.2d 1367, 1379-80, 231 USPQ 81, 90 (Fed. Cir. 1986).

A typical application must conform to screen definition (seen in a standard windows such as resize/zoom) because it is necessary, and not need to be described by a skilled artisan.

As per Applications' mention in remarks p. 16: 1-6, Applicants clearly preempt the standard properties of "screen definition". As being already discussed, a typical application embedded, or within a Microsoft Windows can be resized and zoom (run-time). This feature must conform to or comply with a standard size of a computer screen, provided at compiled time before it becomes a run-time application. This could not be new in the art as seen in a CE Window toolkit (presenting in this action) where the CE Window toolkit shows property windows and allows screen definition parameters to be entered in the compiled time.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of application for patent in the United States.

4. Claims 1-3, 13-15, 25-27 are rejected under 35 U.S.C. 102(b) as being anticipated by Microsoft Corporation (hereinafter: Microsoft) "Microsoft Windows CE Toolkit for Visual Basic 6.0 Guided Tour", MSDN Library, 7-1999.

Given the broadest reasonable interpretation of followed claims in light of the specification.

As per Claim 1: Microsoft discloses,

"*A method for executing application programs, comprising:*

receiving at least one application program in a in a handheld mobile wireless client device (Windows CE is installed and run in a handheld client device. For example, see Figure 5, shows an application name "Project1" created by Visual basic and run in a Handheld PC Pro as a default Device);

activating said at least one application program in said handheld mobile wireless client device (It should be noted that "Project1" is only a default name; User who creates the "Project1" can rename this

application properly (e.g. "NorthwindCustomerInformatiuon"). See Figure 14, a toolkit that provides an application is registered in the file system of a "client device" such as Handheld PC Pro, so that when the "project1" can be carried under this toolkit. See page 4, "Starting the Windows CE project");

instantiating a run-time engine in said handheld mobile wireless client device (See Figure 14, it includes "run", "debug", i.e. "project1" (of Figure 5) can be run or debug by a run-time engine of the Handheld PC Pro); *and*

executing said at least one application program by said run-time engine in said handheld mobile wireless client device to create screen definitions with said at least one application program at run-time as if said screen definitions has been defined at compile time (See Figure 14, it provides a run engine "run" or "debug" as in the manner of compilation when a "new project" completely coded; moreover, the Toolkit included with "Setting the Project Properties" (See page 4) provides screen definition setting).

As per Claim 2: Microsoft discloses,

registering said at least one application program with an operating system of said handheld mobile wireless client device; (Figure 14 includes "File" menu. "File" menu is known as a common feature associated with an operating system of the device and used in standard Microsoft Windows (Windows is also an operating system), where programs/files/applications/ are registered in a hierarchical structure. An example to illustrate for registering is seen in Figures 1-2, or Figure 15);

and displaying an icon configured to represent said at least one application program in response to said registration" (This is referred as screen of the client device implemented with windows CE – See icons within/under "Name" in Figures 1-2).

As per Claim 3:

A method for executing application programs, comprising:

receiving at least one application program in a client device (For example, and application is located in "File" menu which might be downloaded from network, or created by a the toolkit. As seen page 4, "Starting the Windows CE Project", provides an Application program selected from "File" menu, or see Figure 5, "Project1, All Figures 16, 17, 18, represent "Application" already/within developed in the client device);

activating said at least one application program (Figure 14 provides activating an application);

instantiating a run-time engine (Figure 14 includes a run-time engine "Run", or "Debug" menu);

executing said at least one application program by said run-time engine (A project/application created within Figure 14 can be executed by "Run" or "Debug" menu);

registering a process identification corresponding to said activated said at least one application program (It is clearly that a project/application selected from "File" menu will be registered under this run/debug process); *and*

executing a GO method by said run-time engine (It is clearly that if a user select the "Run" menu, the predicated/registered project/application will be run under this engine. A typical example, if a user run the application and see en error, he can select "debug" menu which provide "Go" to run the program for debugging purpose).

As per Claims 13-14: The rejection of these claims is in the same reason as in Claims 1-2. See rationale addressed in Claims 1-2 above. It should be noted that a typical client device such as Handheld PC pro is constructed comprises at least a memory and a processor because of standardization of a computing device.

As per Claim 15: The rejection of the claim is in the same reason as in Claim 3. See rationale addressed in Claim 3. It should be noted that a typical client device such as Handheld PC pro is constructed comprises at least a memory and a processor because of standardization of a computing device.

As per Claims 25-26: The rejection of these claims is in the same reason as in Claims 1-2. See rationale addressed in Claims 1-2 above. It should be noted that a typical client device such as Handheld PC pro is constructed comprises at least a readable storage medium because of standardization of a computing device.

As per Claim 27: The rejection of the claim is in the same reason as in Claim 3. See rationale addressed in Claim 3. It should be noted that a typical client device such as Handheld PC pro is constructed comprises at least a readable storage medium because of standardization of a computing device.

Allowable Subject Matter

5. Allowable subject matter of Claims 4-12,16-24 and 28-36

Claims 4-12,16-24 and 28-36 are allowed because the independent Claims of these Claims are rewritten in independent form including all of the limitations of the base claim and any intervening claims in accordance to Allowable subject matter in the prior action.

Conclusion

6. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Wei Y. Zhen can be reached on (571) 272-3708.

The facsimile number for the organization where this application or proceeding is assigned is the Central Facsimile number **571-273-8300**.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

Ted T. Vo
Primary Examiner
Art Unit 2191
April 28, 2006

Notice of References Cited	Application/Control No. 09/985,880	Applicant(s)/Patent Under Reexamination HAMILTON ET AL.	
	Examiner Ted T. Vo	Art Unit 2191	Page 1 of 1

U.S. PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Name	Classification
	A	US-			
	B	US-			
	C	US-			
	D	US-			
	E	US-			
	F	US-			
	G	US-			
	H	US-			
	I	US-			
	J	US-			
	K	US-			
	L	US-			
	M	US-			

FOREIGN PATENT DOCUMENTS

*		Document Number Country Code-Number-Kind Code	Date MM-YYYY	Country	Name	Classification
	N					
	O					
	P					
	Q					
	R					
	S					
	T					

NON-PATENT DOCUMENTS

*		Include as applicable: Author, Title Date, Publisher, Edition or Volume, Pertinent Pages)
	U	Microsoft Corporation, "Microsoft Windows CE Toolkit for Visual Basic 6.0 Guided Tour", MSDN Library, pages: 1-18, 1999.
	V	
	W	
	X	

*A copy of this reference is not being furnished with this Office action. (See MPEP § 707.05(a).)
Dates in MM-YYYY format are publication dates. Classifications may be US or foreign.

Microsoft Windows CE Toolkit for Visual Basic 6.0 Guided Tour

Microsoft Corporation

July 27, 1999

Summary: This document will take you on a tour of the features contained in the Microsoft® Windows® CE Toolkit for Visual Basic® 6.0. You will learn how to create a simple application that is designed on a traditional Windows desktop and executed on a Windows CE device, and how to access data stored on a SQL ServerÃ‚Â™ 7.0 on a Windows NT® Server. (21 printed pages)

System Requirements

The following software applications must be installed on a machine in order to get the demo running:

- Windows 95, Windows 98, or Windows NT 4.0 (with Service Pack 3).
- Windows CE Services 2.2 or later.
- Microsoft Visual Basic 6.0.
- Windows CE Toolkit for Visual Basic 6.0.
- ADOCE (not necessary for some Jupiter Class machines).

In addition, the following hardware is required to run the demo:

- Base machine for running the above software.
- Windows CE H/PC Pro ("Jupiter" Class).

This demo was written with a Windows NT 4.0 configuration (installed according to the instructions below) and a Sharp Mobilon Tripad H/PC Pro machine.

Getting Started

Setting Up the Operating System and Windows CE Services

Set up the operating system of your choice as you normally would. Install Windows CE Services 2.2. Install all other software.

- Windows NT 4.0
- Windows NT Service Pack 3
- CE Services 2.2
- Windows NT Service Pack 4
- SQL Server 7.0
- Visual Basic 6.0
- Windows CE Toolkit for Visual Basic 6.0

Setting Up the Database

This guided tour requires the Northwind database found in Microsoft Office Professional or Premium Editions. Either Office 97 or Office 2000 will suffice. To find the database, open the "Find" dialog ("Start" -> "Find" -> "Files or FoldersÃ‚Â...", from the Windows Start menu) and search for "Northwind.mdb". Take note of the location of the Northwind database.

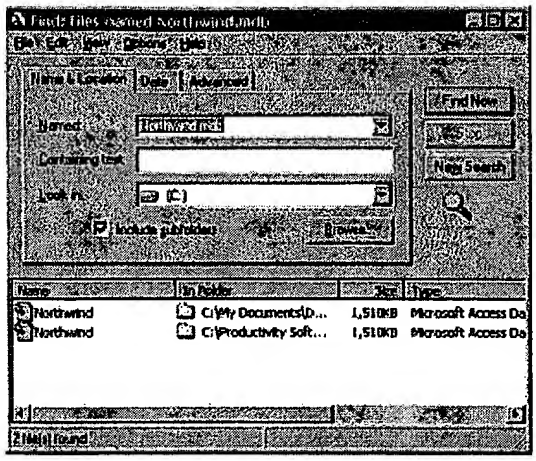


Figure 1. Locating the Northwind.mdb

Copying the Database to the Windows CE Device

With Windows CE Services 2.2, data copied to a Windows CE device will automatically be synchronized with the "master" copy of the data existing in an Access file or in a SQL Server.

To copy data from the parent machine to the Windows CE device, first establish a partnership and connect to the parent machine. For more information on how to do this, consult your Windows CE device documentation.

Once the connection is made, open the Windows Explorer and navigate to the Mobile Devices icon. Click on the connected device and import the required database tables.

- Open the Windows Explorer.
- Navigate to the Mobile Devices folder.
- Click on the connected device to highlight it.
- Select the "Tools" -> "Import Database TablesÃ‚Â..." menu item.

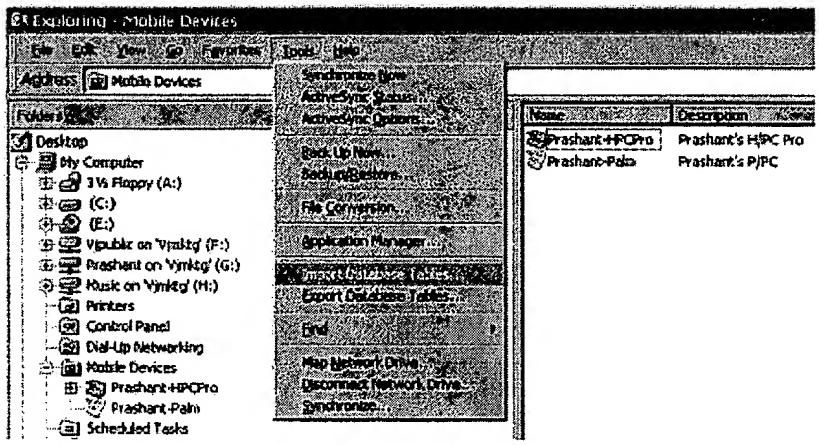


Figure 2. Importing tables from Northwind.mdb

In the resulting dialog, select the Northwind Microsoft Access database you found earlier. Once the database is opened, a list of all tables is shown. After selecting the tables, the information is copied to the device and a synchronization relationship is kept.

- Check off only the tables you wish to keep synchronized between the device and the parent machine.
- Change the location of the database to reflect what is shown in the dialog below.

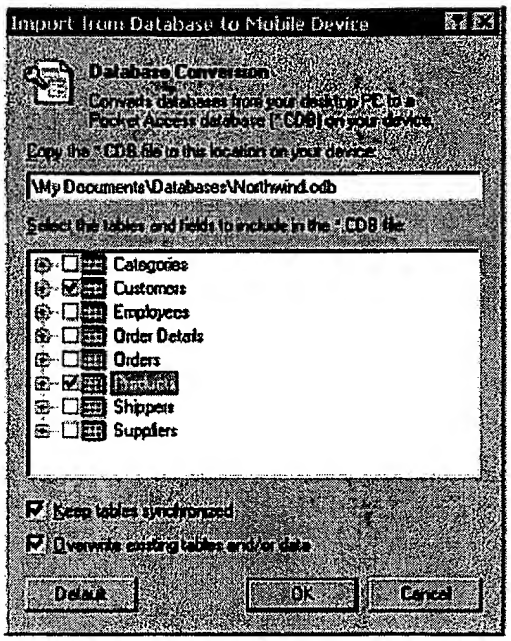


Figure 3. Changing the database location

To turn off database synchronization, simply uncheck "Tables" in the ActiveSync options dialog ("Tools" -> "ActiveSync OptionsÃ‚Â..." from the Mobile Devices folder).

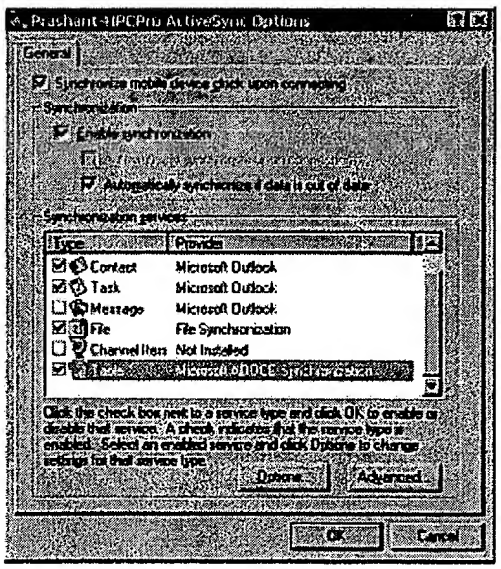


Figure 4. Turning off database synchronization

Benefits of Windows CE Toolkits

With the Windows CE toolkits, developers can use the programming knowledge they have accumulated throughout their careers to build applications for a whole new class of devices and hardware. Windows CE is the next frontier of Windows development and allows businesses to extend their software development to whole new markets previously unavailable to them.

With Windows CE running on Web TV, businesses can build applications for a home consumer market that would otherwise be reticent about using a traditional personal computer. With Windows CE running on an AutoPC, developers can build truly mobile applications for fleets of salespeople, executives on the go, or consumers on the road. With Windows CE running on a handheld PC, developers can expand their opportunities to include non-traditional computer users who want simplicity and ease of management of above all

else. Additionally, with Windows CE running on a palm-sized PC, programmers can take advantage of a data access device that allows anyone, anywhere, to access the information needed to run a business.

Because Windows CE development is as easy as traditional Windows development, these opportunities are available to developers today, with little additional training needed. The options are limitless, and because we are just at the beginning of this new phenomenon, developers have the entire spectrum of application development at their disposal. The "killer application" for Windows CE has yet to be discovered and developers who get in on the ground floor today will have a leg up on the competition in determining just what that killer application is.

This Guided Tour will help you create a complete data-driven application that can be copied and executed on a Windows CE device. For this demonstration, we will be using the Sharp Mobilon Tripad Handheld PC (H/PC) Professional Edition. However, any H/PC Pro will do.

Developing a Windows CE Application

Starting the Windows CE Project

Building a Windows CE application in Visual Basic doesn't involve any unusual steps at all. In fact, let's just go ahead and create a new project and see what it looks like.

- In Visual Basic 6.0, go to the "File" menu and select "New Project".

You'll notice that in the New Project dialog there are several options to choose from. Some of these options are Windows CE Forms. When building a Visual Basic application, all you need to do is select one of these Windows CE Forms, drag and drop components, and add code just as you normally would. There's no special magic or incantations to build a Visual Basic application for a Windows CE device. All of the forms, as well as debugging tools and class libraries, are installed by the Windows CE Toolkit for Visual Basic 6.0. The Toolkit is essentially an add-on to the Visual Basic environment and, in fact, requires Visual Basic 6.0 Professional Edition or greater.

- Select "Windows CE HPC Pro Project" template in the "New" tab.

Setting the Project Properties

Once the template is selected, a project properties dialog will appear. In this dialog, you can specify all the information for how and where the project will be executed on the remote device or emulator. With all the rudimentary elements to application development, including form size, project location, and descriptive elements, the dialog will help you get up and running so that you can get to the important phases of application design and development.

For the purposes of this example, set the project's properties as shown in the dialog below.

- Set the Form Width and Form Height to 640x400.
- Enter text for the Project Description field.
- Change the Remote Path to "*\NorthwindCustomer.vb*".
- Set the Update Components field to "Always".

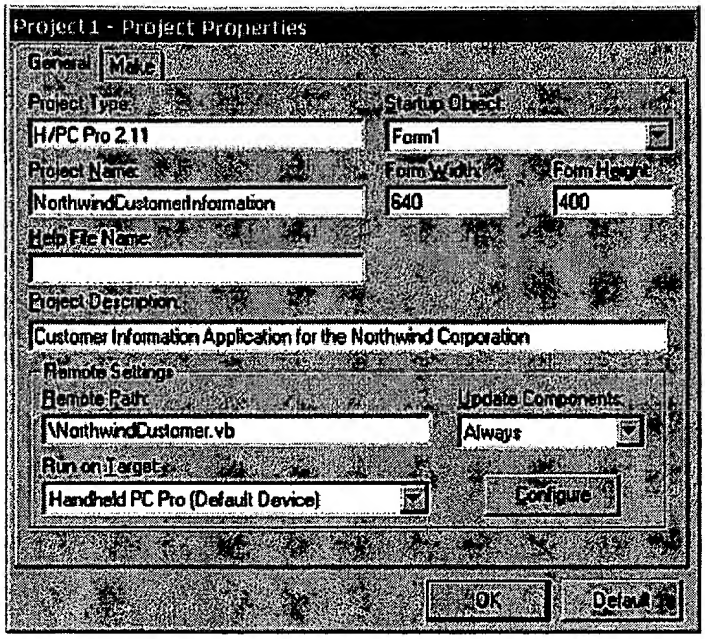


Figure 5. Setting project properties

Press **OK** to move on to designing the form.

Setting Up the Toolbox

After the project properties are finalized, a Visual Basic Form will appear, allowing you to visually design your application. You will start out with a blank form, as shown below.

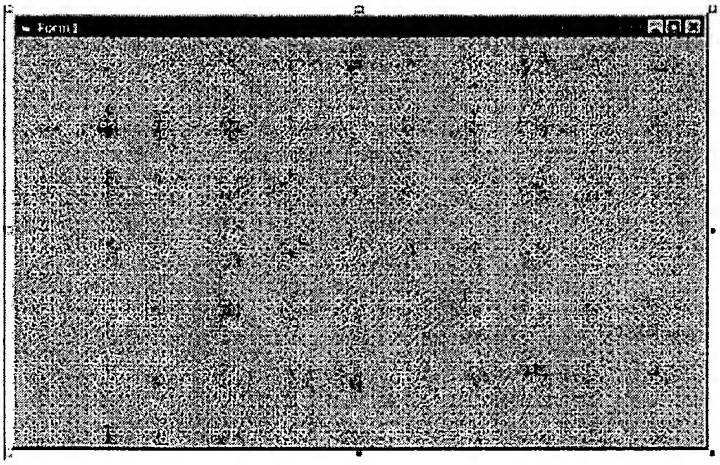


Figure 6. Visual Basic form

Now we will need to include all of the Windows CE-specific controls in our Toolbox by importing them from the on-system libraries.

- Right-click on the Toolbox and select "ComponentsÃ‚Â..."
- In the resulting dialog, select all the components with the words "Microsoft CE" prepended to them:

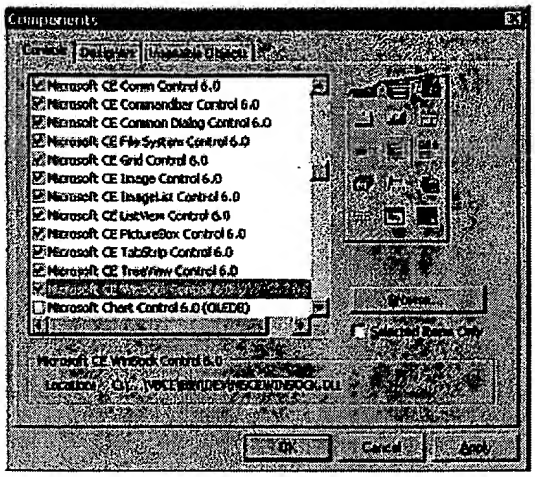


Figure 7. Selecting components for the Toolbox

All of the components will now appear on the Toolbox and can be dragged and dropped from there onto the Form.

Windows CE Design Considerations

Our application will be a simple record lookup form that will allow us to select a customer and gather information about the customer. When designing the Form, we should keep in mind that the application will be run on a small device, with a moderately small screen (640 x 480 resolution). Further, many of the H/PC Pro devices have touch-screens, which can be activated either by a stylus or a touch of the finger. Therefore, we will make sure that all of our fonts are large enough to read and all buttons or sliders will be large enough to be manipulated by a finger, as well as a stylus. It is important to recognize the unique attributes of the devices for which you build applications.

Designing the Form

We will design the Form by first placing a combo box on the Form as shown below.

- Drag and drop a Label control from the Toolbox onto the Form.
- Drag and drop a Combo Box control from the Toolbox onto the Form.
- Position the Combo Box and Label appropriately.
- Set the "Style" property in the ComboBox so that it says "2 - Dropdown List".
- Set the "Name" property on the ComboBox to "comboCustomerName".

Now we will need to flesh out the rest of the Form with Label and TextBox controls.

- Drag and drop Label and TextBox controls from the Toolbox onto the Form.
- Position the controls appropriately.
- Set the TextBox control's **Name** property as shown below:

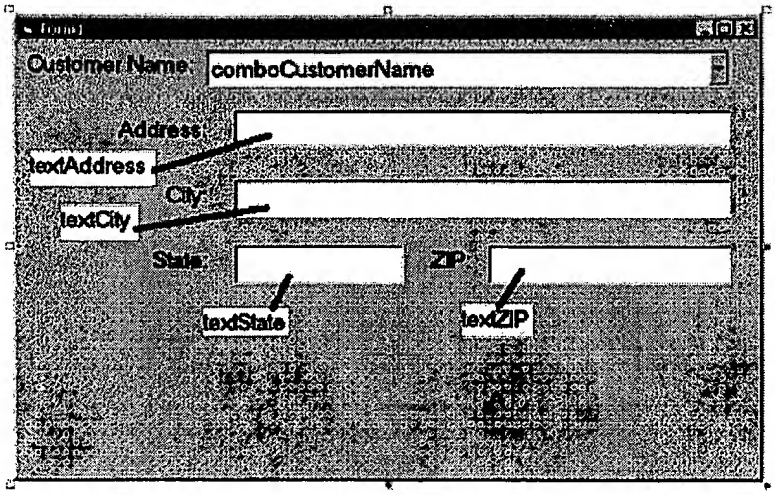


Figure 8. Setting the Name property of the TextBox control

In some cases, you may want to create more complex user interfaces using additional Windows CE ActiveX® controls. However, it is important to realize that not all of the functionality found on traditional Windows desktops is present in the Windows CE operating system.

- Drag and drop a FileListBox control from the Toolbox onto the Form.
- The following dialog box will appear:

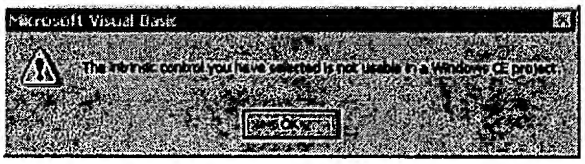


Figure 9. Message for an unavailable control

As you can see, the Windows CE Toolkit for Visual Basic 6.0 helps you by telling when a control is unavailable on the Windows CE operating system. Because the Win32® API present on Windows CE is a subset of that found on traditional Windows desktops, the Windows CE Toolkits help tremendously by informing you of unsupported functionality.

Event Handlers for Windows CE ActiveX Controls

Once the Form is complete, we need to set up an event handler so that we can update the customer information whenever a new customer is selected. One of the beauties of Windows CE development in Visual Basic is that there is very little difference between developing traditional Windows desktops and developing Windows CE applications.

To add an event handler to the Combo Box, go to the Code Editor and select "comboCustomerName" in the "Object" dropdown and "Click" in the "Procedure" dropdown. Visual Basic will automatically jump you to the Visual Basic Code Editor in the **comboCustomerName_Click** function:

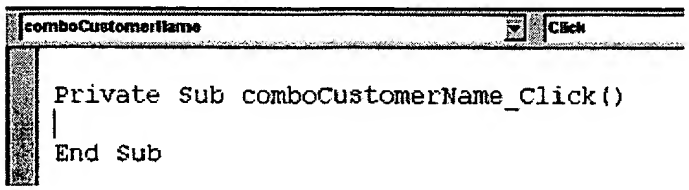


Figure 10. Adding an event handler

Adding Data Access Code

With Windows CE and ActiveX Data Objects for CE (ADOCE), a Windows CE device can contain a local copy of a remote ODBC data source. When the Windows CE device is connected to the "parent" computer, the database tables are automatically synchronized. New information on the database server is automatically copied to the CE device's local store, and any information changed on the CE device is uploaded to the database server. With the obvious management simplicities of the Windows CE device, this is a perfect solution for task-specific applications.

Importing ADO functionality

ADOCE functionality can be found in a separate library that is installed by the Windows CE Toolkit. To import the library into the current project, go to the "Project -> References" menu and look for the "Microsoft CE ADO Control" component.

- Go to "Project -> References".
- Select "Microsoft CE ADO Control (ADOCE) 2.0".

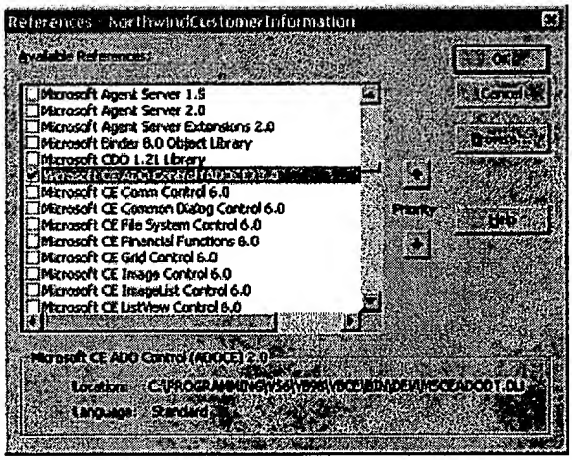


Figure 11. Importing ADOCE functionality

Setting up data access

Once again, the programming models in developing applications for the traditional Windows desktop and the Windows CE device are very similar. We've already seen how building Visual Basic Form-based applications requires little additional training. Similarly, developing ADO applications for Windows CE is no more difficult than programming desktop applications.

We must first add a few global variables. The first two are strings, one representing the SQL statement that we will execute on the database, the other the physical location of the database on the device itself:

```
' the SQL string and database location
Const SQL = "SELECT * FROM CUSTOMERS"
Const DBName = "\My Documents\Databases\Northwind.cdb"
```

Once the global variables are added, we need to create a global variable representing the Recordset that we will manipulate. Add the following line of code:

```
' the Recordset for the Customer table
Dim rsCustomers as ADOCERecordset
```

Notice how Visual Basic's IntelliSense® technology continues to work, even when creating a Windows CE project. That's because the Windows CE Toolkit is an add-in to the traditional Visual Basic product, allowing you to take advantage of all the features in the world's most popular development environment while you build applications for Windows CE.

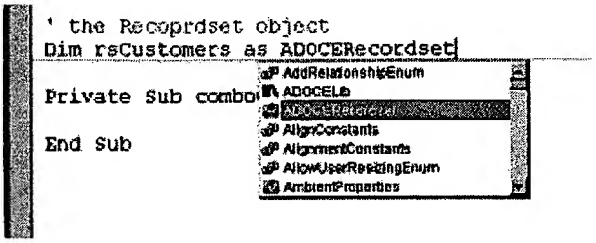


Figure 12. Setting up data access

Opening the database

Opening the database is as easy as it is in desktop development. First, we need to create the **Form_Load()** function. You can either hand-code the function or go to the "Object" dropdown and select "Form".

Once the **Form_Load()** function is available, allocate the **Recordset** object by adding the following lines of code inside the function:

```
Private Sub Form_Load()

    ' create the recordset object
    Set rsCustomers = CreateObject("adoce.recordset")

    ' fill the customers list
    FillCustomers

End Sub
```

The **FillCustomers** function is something we will create in a moment to populate the data in the combo box control.

Now we need to open the **Recordset** itself so that information can be extracted from it. Change to **Form_Load()** function to include the new line of code:

```
Private Sub Form_Load()

    ' create the recordset object
    Set rsCustomers = CreateObject("adoce.recordset")

    ' open the table into the recordset
    rsCustomers.Open SQL, DBName, 1, 3, 1

    ' fill the customers list
    FillCustomers

End Sub
```

The **Open** method on the **Recordset** object accepts five parameters: the SQL string (which we already declared as a constant), the Database name and location (which we already declared as a constant), and three integers representing read/write capability and more. For more information on the ADO "Open" command, consult the ADO for Windows CE documentation.

Populating the ComboBox

We will need to create a **FillCustomers** function that will be called by the Form_Load() subroutine. The **FillCustomers** function is responsible for traversing the rows in the Customers table and extracting the appropriate information to place in the comboCustomerName dropdown.

The first thing we need to do in the **FillCustomers** function is to move to the first element in the **Recordset**. Next, we need to create a while loop that will help us traverse the **Recordset** with the end-of-file state as the condition:

```
Private Sub FillCustomers()

    ' move to the first element in the recordset
    rsCustomers.moveFirst

    ' create a while loop to fill the combo box
    Do While Not rsCustomers.EOF

    Loop

End Sub
```

Inside the while loop we need to add the name of the customer to the combo box and then move on to the next row in the **Recordset**:

```
Private Sub FillCustomers()

    ' move to the first element in the recordset
    rsCustomers.moveFirst

    ' create a while loop to fill the combo box
    Do While Not rsCustomers.EOF

        ' add the last name to the combo box
        comboCustomerName.AddItem
(rsCustomers.Fields("ContactName").Value)

        ' move to the next item
        rsCustomers.moveNext

    Loop

End Sub
```

Responding to a ComboBox event

Once a customer name is selected from the combo box, the "click" subroutine will automatically be called. Within the "changed" subroutine, we will need to search the Recordset for the name selected in the combo box. Once the record containing the name is found, we can use the rest of the data in the row to fill out the remainder of the Form.

First, we need to move to the first item in the Recordset. Next, we must once again create a while loop and add some code to search the tables for the specified name. When the name is found, we should break out of the loop. Otherwise, we should continue with the loop.

```
Private Sub comboCustomerName_Click()

    ' move to the first element in the Recordset
    rsCustomers.MoveFirst

    ' search for the customer name
    Do While (Not rsCustomers.EOF)

        If (rsCustomers.Fields("ContactName").Value <>
comboCustomerName.Text) Then

            ' move to the next item in the Recordset
            rsCustomers.MoveNext

        Else

            ' get the heck out of the loop
            Exit Do

        End If

    Loop

End Sub
```

Updating the text fields

To complete the application, we must add code to update the text fields on the Form with the corresponding data in the Customers table. After the end of the loop in the comboCustomerName_Click function, we need to add the following four lines of code:

```
textAddress.Text = rsCustomers.Fields("Address").Value
textCity.Text  = rsCustomers.Fields("City").Value
textState.Text = rsCustomers.Fields("Region").Value
textZIP.Text = rsCustomers.Fields("PostalCode").Value
```

Deploying a Windows CE Application

After we are finished developing the Windows CE application, we must copy it to the remote device and execute it. With the Windows CE Toolkit for Visual Basic 6.0, deploying a Windows CE application is as easy as pressing the "start" button. The Toolkit will automatically copy the application (and any needed DLLs such as Tab Strip controls and more) to the device. Further, it will launch the application on the device for us.

Running the Application

Press the **Start** button on the Visual Basic main menu.

- Press the **Start** button.

After doing so, the Visual Basic run time, the necessary DLLs, and any controls that have yet to be installed on the device are automatically copied.

After that process is complete, the application itself is copied.

The application should automatically start on the device.

Using the Windows CE Accessories

Several Windows CE tools that enable you to go the distance with Windows CE software development are installed alongside the Windows CE Toolkits. The Windows CE Remote Zoomin application, for example, lets you take screenshots of a application as it runs on a remote device.

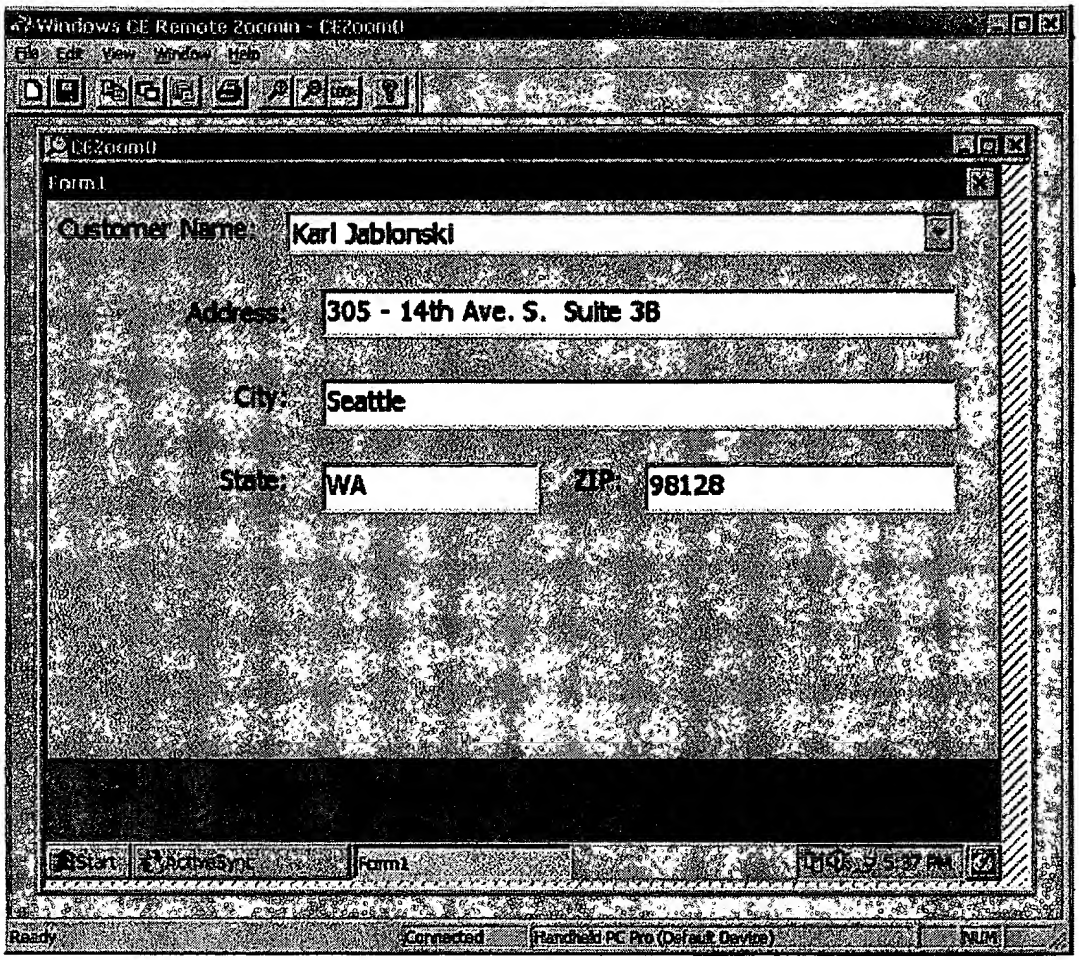


Figure 13. Windows CE Remote Zoomin

Debugging a Windows CE Application

The debugger in the Windows CE Toolkit for Visual Basic 6.0 can be started by selecting the "Go" item from the "Debug" menu.

- Select "Debug->Go" from the main menu.

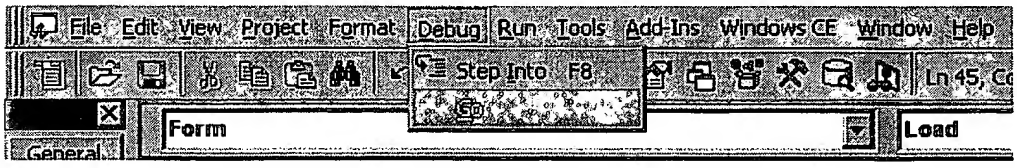


Figure 14. Starting the debugger

The Windows CE Toolkit for Visual Basic 6.0 Debugger

The debugger is a separate program that is launched after the "Go" menu item is selected. From within the debugger, you can set breakpoints, drag variables to the watch window, inspect the callstack, and more.

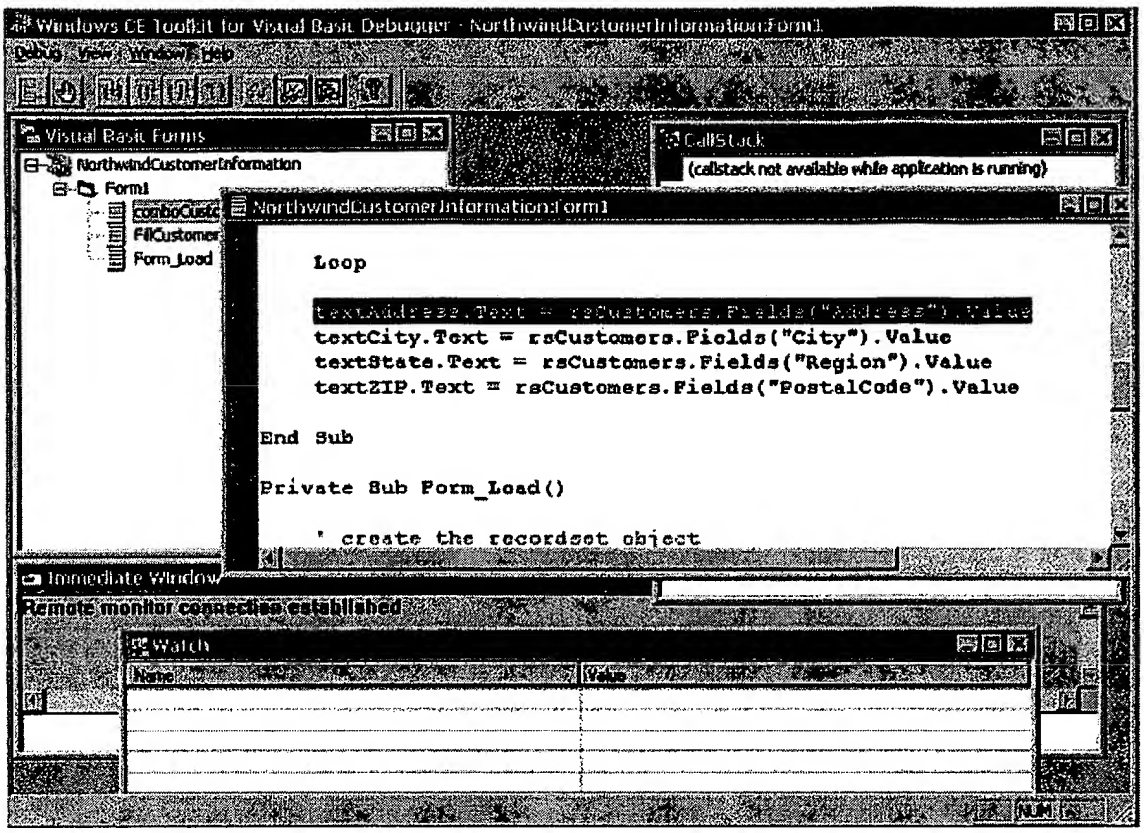


Figure 15. Inside the debugger

Setting a Breakpoint

In order to set a breakpoint, you need to view the code inside the debugger application. In this release, setting a breakpoint in the Visual Basic code editor will not affect operation of the Windows CE debugger.

- Open the "Form1" node in the Visual Basic Forms window.
- Double-click on the comboCustomerName_Click item.
- The Code Editor will automatically launch.
- Click in the gutter of the editor next to the line highlighted in the picture above.

More Complex User Interface Tasks

There are several more ActiveX controls for user interface design including tab controls, scrollbars, picture boxes, tree controls, and more. To use these controls, simply drag and drop them from the toolbox onto the Form.

Creating Tabbed Pages

In order to use the TabStrip control, you will need to create separate borderless Frame controls. When a specific tab is selected, you need to hide all Frame controls except the one corresponding to the selected tab.

Setting up the TabStrip control

For example, if we were to drag and drop a TabStrip control and add three tabs to it: "New Order", "Order History", and "Web", we would need to add three corresponding Frame controls for each: fraNewOrder, fraOrderHistory, and fraWeb.

- Drag and drop a TabStrip control onto the Form, as shown below:

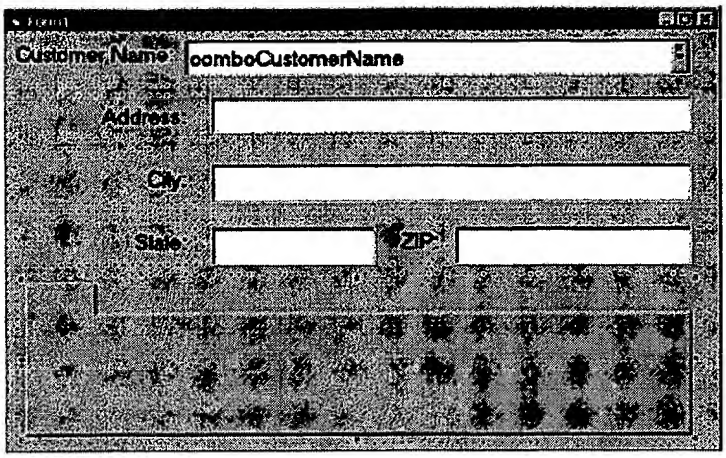


Figure 16. Dragging a TabStrip control onto the Form

- Right click on the TabStrip control and select "Properties"
- Add three tabs and name them "New Order", "Order History", and "Web"

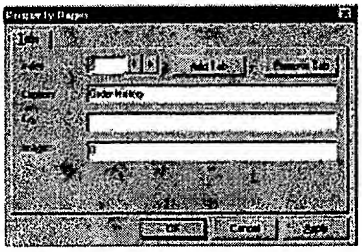


Figure 17. Adding tabs to the control

Creating the tabbed pages

Now you need to drag and drop some Frame controls onto the Form, make them borderless, and position them on the Tab control.

- Drag and drop a Frame control onto the Form. Be sure to drop it onto the Form and not another Frame control.
- Go to the Frame control's property sheet and set its **BorderStyle** property to "0 - None".
- Resize the Frame control as shown below:

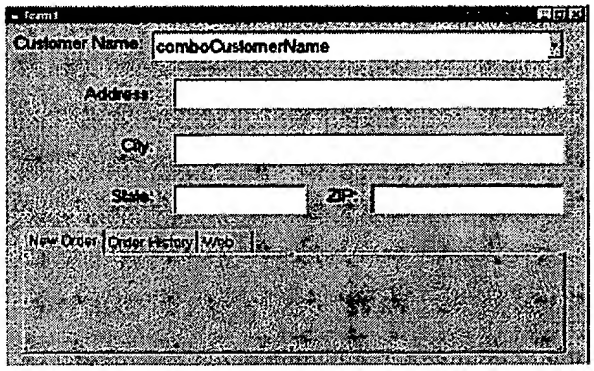


Figure 18. Resizing the Frame control

- Name the Frame control "fraNewOrder".
- Repeat two more times for controls named "fraOrderHistory" and "fraWeb".

You can drag and drop controls onto the individual Frame controls. The controls will become "children" of the Frame control, so whenever the Frame control is hidden, the controls placed on it will be hidden as well. In order to select a specific Frame, go to the Properties sheet and simply select it in the controls list dropdown:

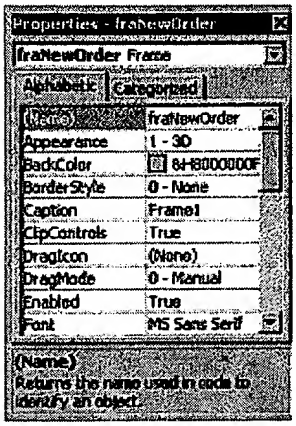


Figure 19. Selecting a frame

Because all the Frames overlap one another, you will need to use the "Order" menu item (shown below as part of the "Format" menu) to bring a Frame to the foreground so that you can drag and drop controls onto it.

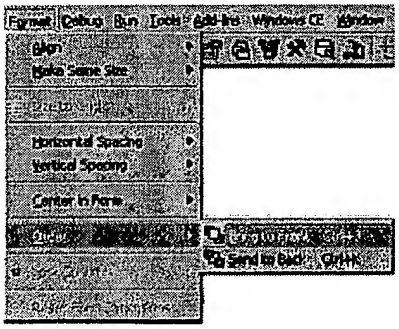


Figure 20. Bringing a frame to the foreground

Setting up tab events

Now that the frames are on the Form, we need to set up the logic so that when a specific tab is selected, the appropriate Frame control is shown. All we need to do is add some code into the TabStrip control's click method that determines what tab was selected and shows the correct Frame control.

- Go to the Code Editor and select "TabStrip1" from the "Object" dropdown.
- The **TabStrip1_Click** method is automatically created for you.
- Add code at the beginning of the function to hide all of the Frame controls:

```
Private Sub TabStrip1_Click()

    fraNewOrder.Visible = False
    fraOrderHistory.Visible = False
    fraWeb.Visible = False

End Sub
```

- Add the following code to parse the TabStrip control's click event:

```
Private Sub TabStrip1_Click()

    fraNewOrder.Visible = False
    fraOrderHistory.Visible = False
    fraWeb.Visible = False

    If (TabStrip1.SelectedItem.Caption = "New Order") Then

    ElseIf (TabStrip1.SelectedItem.Caption = "Order History") Then

    ElseIf (TabStrip1.SelectedItem.Caption = "Web") Then

    End If

End Sub
```

- Now for each If statement, show the correct Frame control:

```
Private Sub TabStrip1_Click()

    fraNewOrder.Visible = False
    fraOrderHistory.Visible = False
    fraWeb.Visible = False

    If (TabStrip1.SelectedItem.Caption = "New Order") Then

        fraNewOrder.Visible = True

    ElseIf (TabStrip1.SelectedItem.Caption = "Order History") Then

        fraOrderHistory.Visible = True

    ElseIf (TabStrip1.SelectedItem.Caption = "Web") Then

        fraWeb.Visible = True

    End If

End Sub
```

The CommandBar Control

Windows CE introduces a new kind of user interface element called a CommandBar. The CommandBar is a special menu bar that hosts not only traditional menu items, but command buttons and combo boxes as well. The CommandBar control replaces the main title bar of the Form.

Using the CommandBar control

Since the CommandBar control replaces the Form's title bar, we will need to remove all borders from the Frame itself. Then, simply drag and drop the CommandBar control onto the Form. While the control can be placed anywhere on the Form, when the application is run, the CommandBar will become the new title bar for the Form.

- Select the main Form.
- Go to the Properties sheet and set the "Border" property to "0 - None".
- Drag and drop the CommandBar control onto the Form.

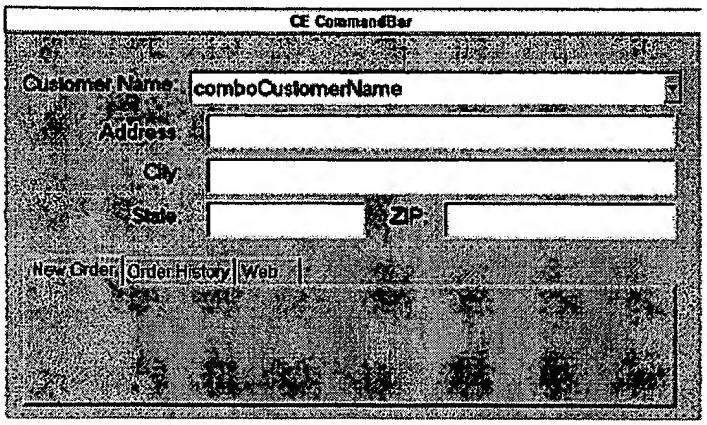


Figure 21. Dragging and dropping the CommandBar control onto the form

- Go to the **Form_Load** function and add a line of code to set the width of the CommandBar control:

```
Private Sub Form_Load()

    ' create the recordset object
    Set rsCustomers = CreateObject("adoce.recordset")

    ' open the table into the recordset
    rsCustomers.Open SQL, DBName, 1, 3, 1

    ' fill the customers list
    FillCustomers

    ' set the toolbar's width to be the width of the application
    CommandBar1.Width = Me.Width

End Sub
```

Adding elements to the CommandBar control

To add controls to the CommandBar, use the **Add** method of the CommandBarControls collection. Once you add member controls to the CommandBar control, the CommandBar reacts to user input by generating events. A CommandBar control generates a specific event for each type of member control. For example, when a user selects any **CommandBarButton** object, the control fires the ButtonClick event. You therefore must code the event to determine which particular **CommandBarButton** object was selected by the user.

You can add the following types of object to your CommandBar control:

- CommandBarButton
- CommandBarComboBox
- CommandBarMenuBar

For example, adding a menu to the CommandBar control involves first creating the CommandBarMenuBar object and then adding it to the Controls collection of the CommandBar control.

```
Dim m As CommandBarMenuBar
Set m = CommandBar1.Controls.Add(cbrMenuBar)
```

Then, adding the actual menus is as easy as creating the menu object, and adding elements to it.

```
Dim mnuFile
Set mnuFile = m.Items.Add(, "mnuFile", "File")
mnuFile.SubItems.Add , "nmuNew", "New"
mnuFile.SubItems.Add , "nmuOpen", "Open"
mnuFile.SubItems.Add , "nmuSave", "Save"
```

The Windows CE Toolkit for Visual Basic 6.0 documentation includes more sample code for using the CommandBar control.

Handling CommandBar events

There is no specific event handler for the individual menu items. Instead, you must use the MenuClick event handler of the CommandBar control and parse the **Key** property for the correct sub-menu item:

```
Private Sub CommandBar1_MenuClick(ByVal Item As
CommandbarLib.Item)

    If (Item.Key = "nmuNew") Then

        MsgBox "The New item was selected"

    End If

End Sub
```

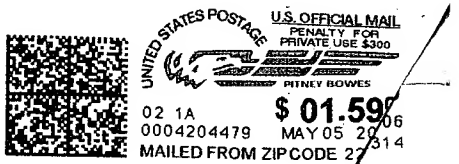
Summary

Using the Windows CE Toolkit for Visual Basic 6.0, you can take advantage of the world's most popular development tool and use your existing skills to build and deploy compelling applications for the next generation of computing devices.

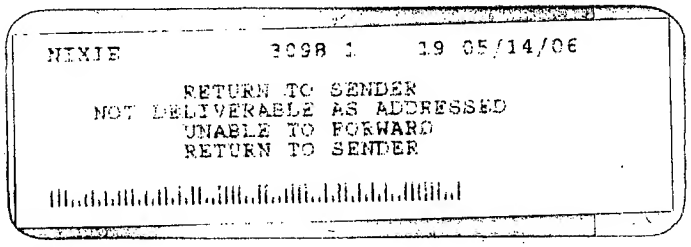
Organization TC2100 Bldg./Room RANDOLPH

U. S. DEPARTMENT OF COMMERCE
COMMISSIONER FOR PATENTS
P.O. BOX 1450
ALEXANDRIA, VA 22313-1450
IF UNDELIVERABLE RETURN IN TEN DAYS

OFFICIAL BUSINESS



AN EQUAL OPPORTUNITY EMPLOYER



UTF
4/12